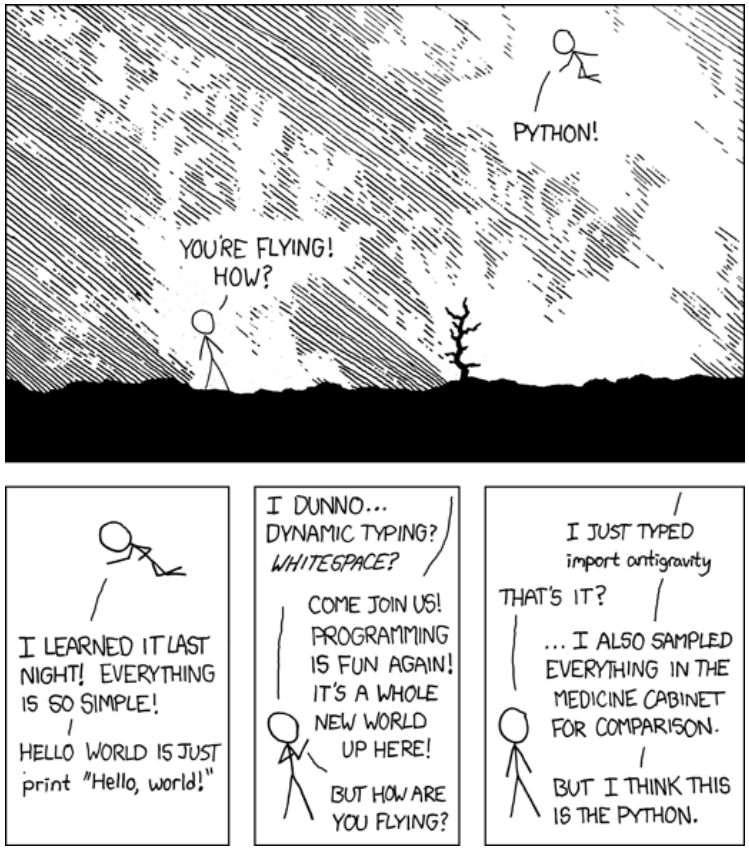
Introduction & setup

Practical Programming in Chemistry

Prof. Philippe Schwaller

Welcome to Practical Programming for Chemists!



XKCD

Not your typical course.

But a course, many chemists and I, would have loved to have during their Bachelor.

This course is all about how to talk (or write) to and use your computer...

So, it gets things done in chemistry, more efficiently.

Whether you go into computational chemistry, or not, you will produce data, and programming skills make a difference.

This is not what your friends in computer science learn.

It is much more practical.

Note: You will most likely learn more during the exercises and project, than the lecture.

Grew up in Fribourg, Switzerland
- French
- Swiss German / German

NCCR Catalysis

TT Assistant Professor
in Digital Chemistry
since Feb 2022

EPFL LIAC

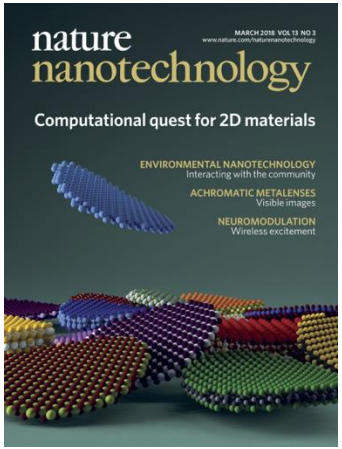
EPFL

MANCHESTER 1824
The University of Manchester

UNIVERSITÄT BERN

PhD in Chemistry
and Molecular Sciences (‘21)
Prof Jean-Louis Reymond

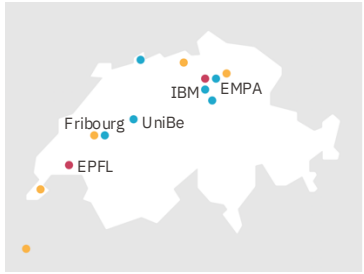
Closing the loop…



Materials Science &
Engineering

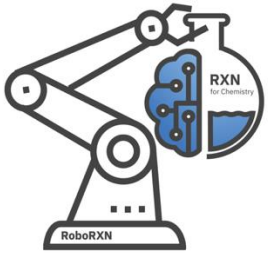
Virtual screening &
simulation workflows
Prof Nicola Marzari

BSc (‘14)/MSc (‘16)



MPhil in Physics (’19)
Dr Alpha Lee

UNIVERSITY OF CAMBRIDGE



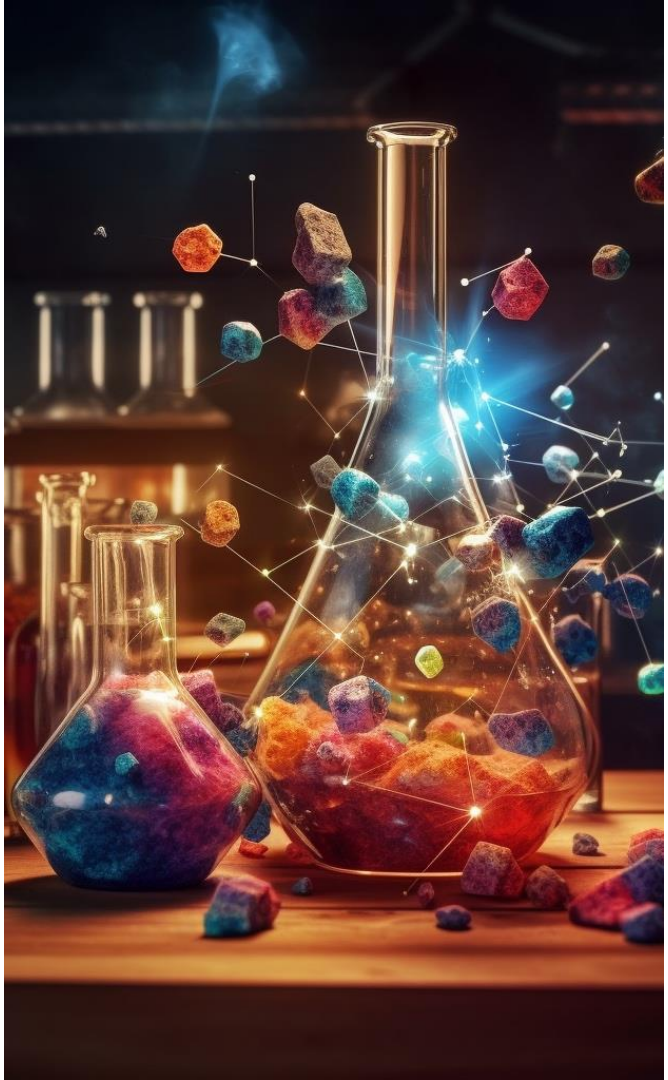
Machine learning for chemical synthesis
Intern/PhD/Postdoc
Dr Teodoro Laino

Empa
Materials Science and Technology

Lab work on ternary polymer
blends for organic solar cells
Prof Frank Nüesch

IBM **Research** Europe

Mission:
Accelerating Discovery with Digital Chemistry



LIAC Team

PhD students
Bojana Ranković
Oliver Schilter (IBM Research)
Andres CM Bran
Junwu Chen
Jeff Guo
Victor Sabanza Gil (Luterbacher)
Paulo Neves (Janssen)
Rebecca Neeser (Correia, VantAI)
Sarina Kopf (Nevado)
Daniel Armstrong
Joshua Sin (Roche)
Sandro Agostini (IBM Research)
Théo Neukomm (Intel/Merck)
Salomé Guilbert (Röthlisberger)
Matt Hart (Trospha)

Rebecca

Sarina

Daniel

Admin
Annick Delmonaco

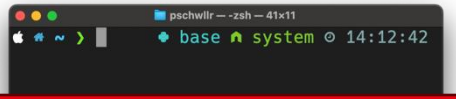
Postdocs/Engineers
Zlatko Jončev
Julian Götz
Edvin Fako
Sergey Pozdnyakov
Jeremy Goumaz

Project students
Shai Pranesh
Antonio Donez
Octavian Susanu
Arne Beckmann
Xuan Vu Nguyen
David Segura

>15 nationalities — one team!
https://schwallergroup.github.io

Funding:
NCCR Catalysis
Swiss National Science Foundation
IBM Research
janssen
VANTAI
Roche
intel
MERCK

The goal for this week – setup your computer



Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage
in **bash**

Visual Studio Code

A better text editor
to handle Python code and everything around it.

Install VS Code as interactive development environment

General purpose programming language.
Create scripts, programs, webapps, …

ANACONDA®

Install Python through anaconda
(main course programming language)

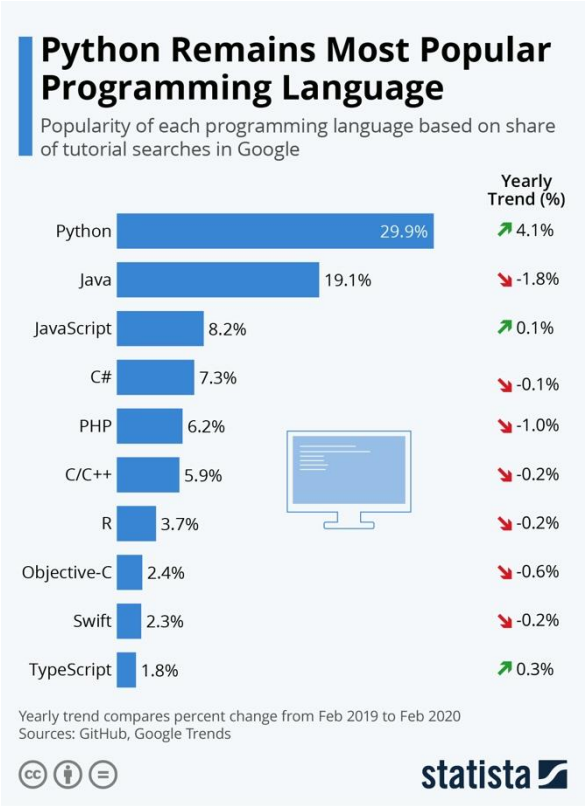
A platform to share, track, and store code.

GitHub

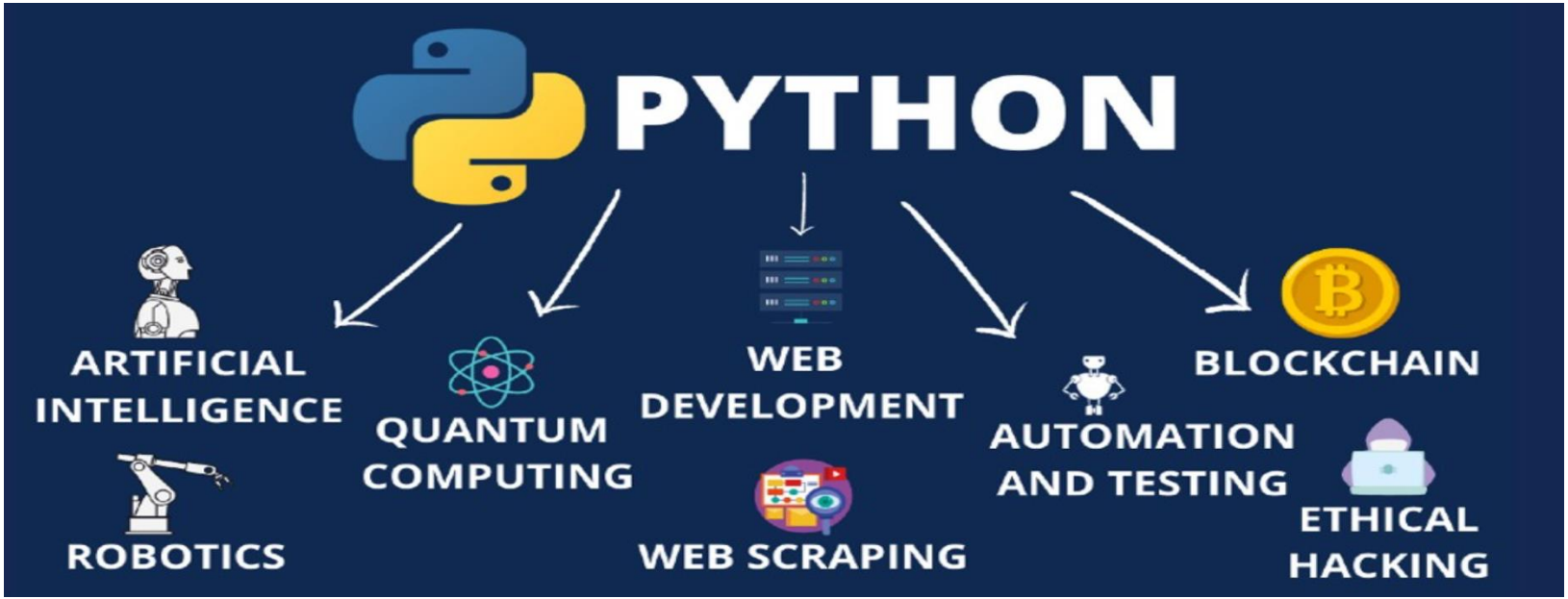
Creating a GitHub account
and getting familiar with basic commands

Main course programming language → Python

- Builds up on what you've learned last year
- Almost like writing/reading in English
- Versatile (data analysis, webpages, automation, visualisation, ...)
- Growing popularity in science communities
- Good for both simple and advanced scripting and applications



Python versatility



https://www.digiprima.com/files/wytrvqctb8ziq4e9icw2/python%20blog.jpg

Almost everything you can imagine is just a pip/conda install away.

Note: *pip install* is one of the commands you can use to install more functionality.

Compared to BA2 course

- Not only Python but the surrounding ecosystem (today you'll learn about bash)
- Focus on examples in chemistry (RDKit and other chemistry modules)
- Not only Python scripting,
  but how to make packages/small applications/…
- How to document and test
- How to use GitHub
- Write in Markdown
- Develop small applications to show your work to others (!!)
- Make Python packages that can be reused (!!)

Organisation

- Slides on moodle
- Exercises: https://github.com/schwallergroup/practical-programming-in-chemistry-exercises
- Bring your own laptop!

Plan for this year

- Two courses per week for 6 weeks, 1 course in week 7, and the final project presentation in week 14.

- CH-200 Practical programming in Chemistry (total 14 lectures)
  - Wednesdays 10h-13h: 19. February - 2. April + 28. May in MED 0 1418
  - Thurdays 14h-17h: 20. February - 27. March in MEB 331

  - Office hours for the remaining of the semester 13h-14h in CH J2

- CH-343 Spectroscopy (total 13 lectures)
  - Wednesdays 10h-13h: 9. April - 21. May
  - Thurdays 14h-17h: 3. April - 22. May

Exercise schedule

Lecture	Topic	Exercise Link
01	Setup your environment	Lecture01
02	GitHub and creating first repositories	Lecture02
03	Conda, Jupyter notebooks, and Python basics	Lecture03
04	Advanced Python: file I/O, functions, error handling, and classes.	Lecture04
05	Numerical operations, data handling, data visualization: `numpy`, `pandas`, `matplotlib`	Lecture05
06	`RDKit` (part I): Reading/Writing, Descriptors, Fingerprints	Lecture06
07	`RDKit` (part II): Substructure matching, Conformer generation	Lecture07
08	Making a Python package	Lecture08
09	Data Acquisition and Cleaning, Web APIs	Lecture09
10	More packaging; project templates, code testing and coverage.	Lecture10
11	Visualization and analysis of chemical data (clustering)	Lecture11
12	Streamlit	Lecture12
13		

The exercises are where you will learn the most (!), and give you plenty of useful skills for the projects.

Evaluation

- Project-based
- The projects are evaluated along three axis
  - Code (40%)
  - Report (30%)
  - Presentation (30%)
- Groups of 2-3 students

- Multiple people working on the same code, presenting your work.

Project examples from last year

Note: Some of the best grades went to students with limited experience at the beginning of the class.

ProtaVision

Welcome to ProtaVision: Delving into Protein Structures and Functions🧬🔬🧪🫓🦠

Welcome to this interactive Jupyter Notebook! 🎉

About This Notebook: This notebook is dedicated to facilitating sequence alignment, a fundamental principle in bioinformatics. Using a custom package, you can input two proteins from a dictionary and efficiently determine the number of matching amino acids, as well as identify the indices of those that do not match. Whether you're delving into bioinformatics for the first time or seeking a streamlined tool for protein analysis, this notebook provides a comprehensive solution.

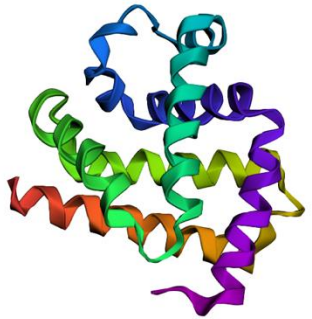
Getting Started: To get started, simply navigate through the sections using the table of contents provided on the ReadMe. Each section is packed with explanations, code snippets, and visualizations to help you grasp concepts effectively.

How to Use: Feel free to experiment with the code! Modify variables, tweak parameters, and run cells to see how different changes affect the outcomes. Don't hesitate to ask questions or seek clarification whenever needed.

Additional Resources: If you're new to Jupyter Notebooks or need a refresher, check out the Jupyter Notebook Documentation for comprehensive guidance.

Questions? If you have any questions, please contact nichelle.sequeira@epfl.ch or aurelie.masson@epfl.ch

Let's Dive In! Start exploring and discovering insights within this notebook. Happy coding! 🚀



https://github.com/nichellesequeira/ProtaVision

rfpred

rfpred 🧪

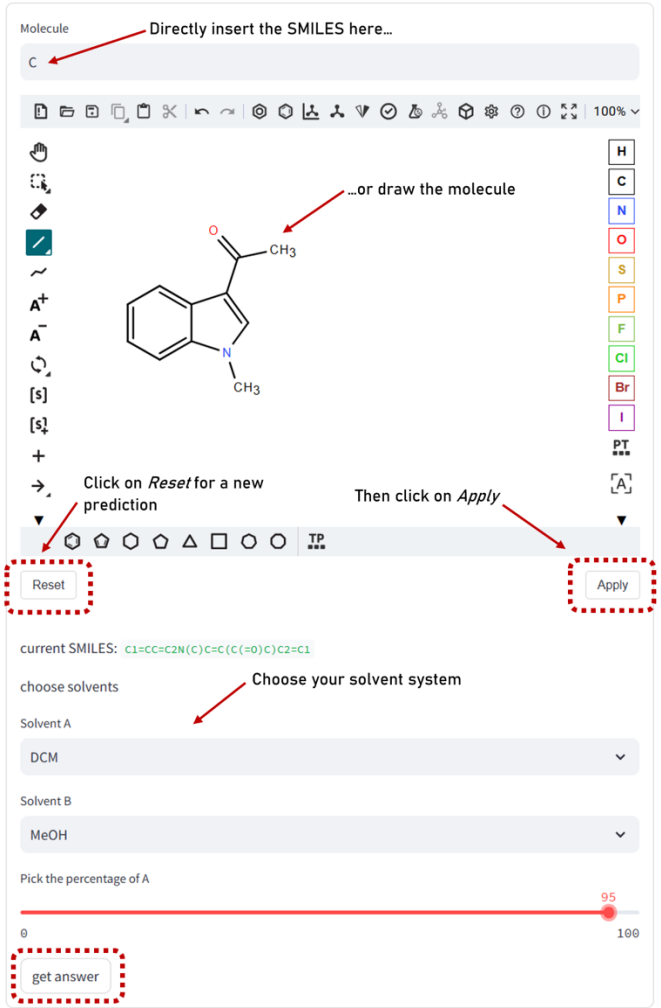
🔍 Package that predicts the Rf values for silica thin-layer-chromatographies (TLCs) in synthesis labs based on the chemical compound and the solvent system.

🔥 Usage

After installing, you should create a new file and activate the environment where you installed ***rfpred***. The function below will start a streamlit web app as a local host in your browser with the prediction interface.

```
import rfpred

# One line to rule them all
rfpred.App.run()
```

https://github.com/MW21P/rfpred



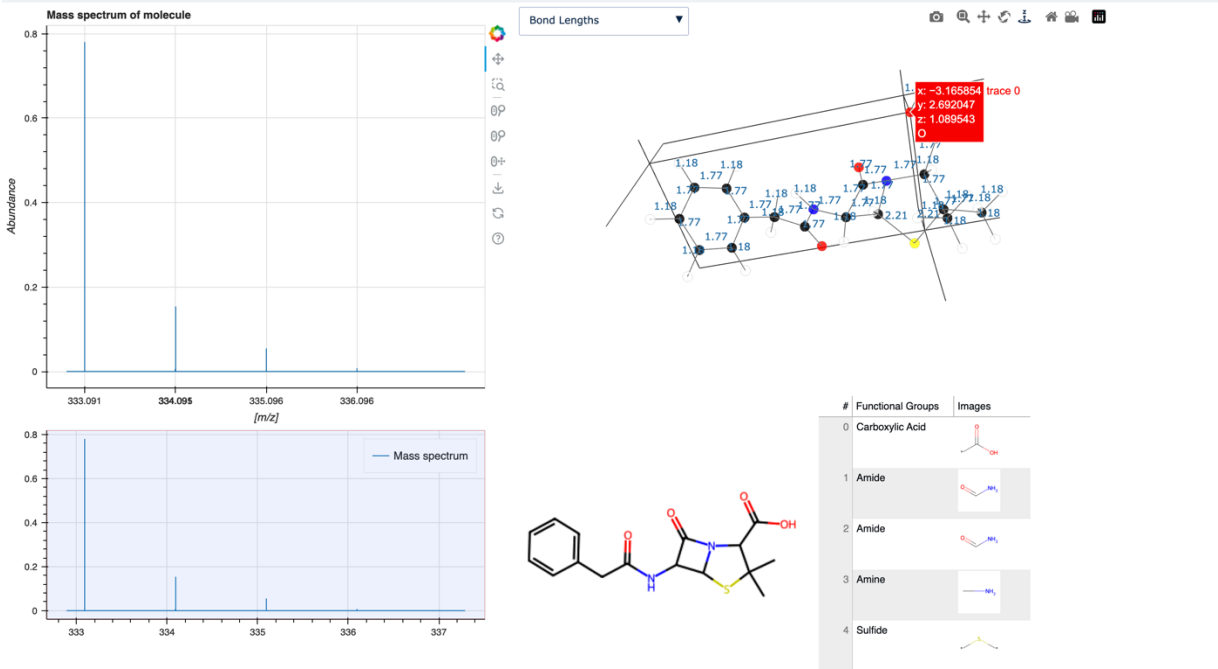
MassiveChem

Maintained? yes | Python 3.10 | Contributors 3 | License MIT | EPFL CH200

- Python package for applied analytical chemistry focused primarily on mass spectrom

Project within *practical programming in chemistry* course -- EPFL CH-200

MASSIVEChem, which stands for "Mass Analytical Spectrometry System for Investigation and Visual Extrapolation in Chemistry", is a pip-installable python package developped at EPFL in 2024 focused on, as its name would suggest, analytical chemistry. The aim of this package is to provide the user with functions in order to simulate the mass spectrum of a molecule and to display this spectrum on a graph. The package also provides other features that can facilitate the chemical analysis of a molecule such as a functional group finder and an unsaturation calculator.



https://github.com/ThomasCsson/MASSIVEChem

Moser.py

MOSER.py (for **M**olecular **O**perations and **S**olutions for **E**quilibria and **R**eactions) is a versatile Python package designed for physical chemistry calculations. In honor of Pr. Jacques-Edouard Moser, whose course "Équilibres et Réactivités Chimiques" at EPFL from 2005 to 2022 inspired the creation of this Python package.

🔬 Features

Unlocking an abundance of analytical tools, **MOSER.py** serves as an invaluable companion for chemistry students, providing intuitive functionalities for essential calculations and analyses. **MOSER.py** streamlines complex tasks, empowering students to deepen their understanding of physical chemistry concepts and accelerate their learning journey. Among other features, it enables the users to:

1. Balance Chemical Equations
Easily balance chemical equations with a simple function call. **MOSER.py** uses advanced algorithms to ensure accurate balancing of equations.

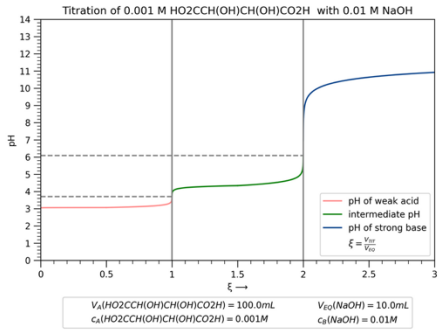
2. Calculate Molar Mass
Determine the molar mass of chemical compounds effortlessly with **MOSER.py**. Input the chemical formula, and **MOSER.py** will compute the molar mass for you.

3. Determine the Reactional Quotient
Evaluate reaction quotients and kinetics parameters using **MOSER.py**'s intuitive interface.

4. Evaluate a Solution's Concentration and pH
Quickly calculate the concentration of solutions and pH values using **MOSER.py**. Specify the relevant parameters, such as the initial concentration of the solute and the volume of the solution, to obtain accurate results.

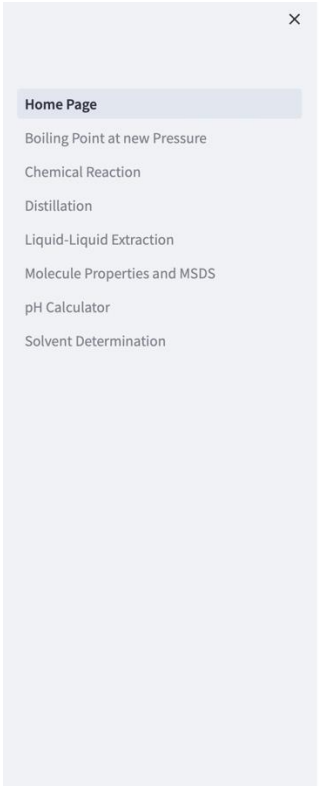
5. Simualte Titration Curves
Trace titration curves for acid-base titrations and visualize the pH changes throughout the titration process. Specify the concentrations of the acid or base, as well as the volume of titrant added, to generate detailed titration curves.

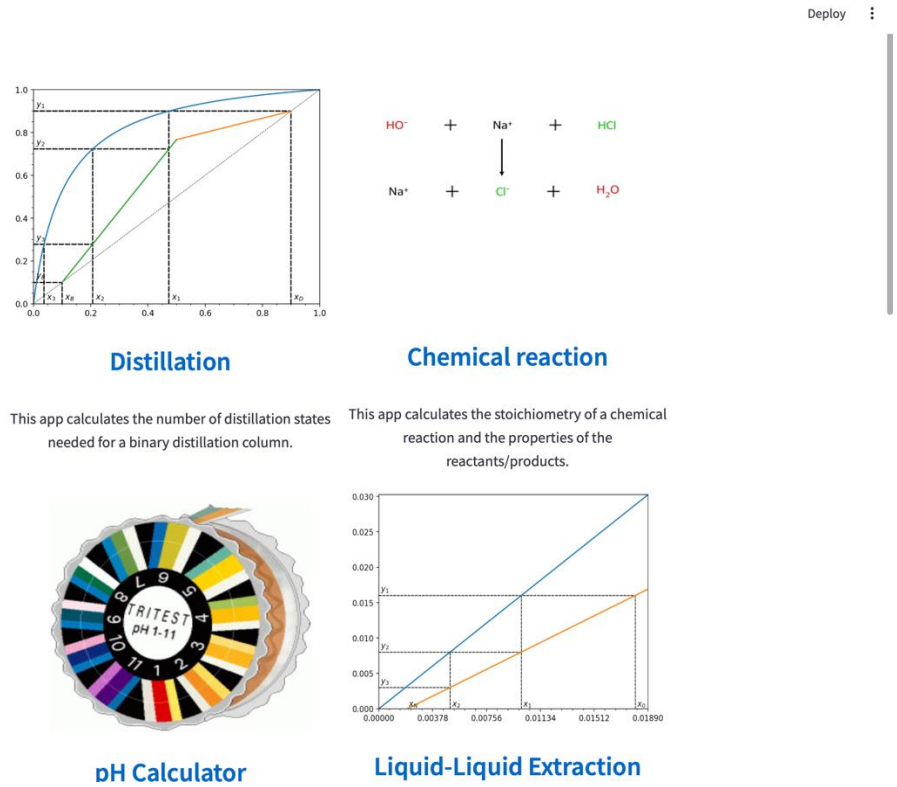
```
from Titration import TitrationUI
TitrationUI()

Select the type of titration you want to perform.
◦ acid titrated by (strong) base --- 1
◦ base titrated by (strong) acid --- 2
Make your choice (1/2): 1

Please enter your data:
◦ chemical formula of titrated acid: HO2CCH(OH)CH(OH)CO2H
◦ (acid) pKa1 [-] = 3.07
◦ (acid) pKa2 [-] (type '*' if there is none) =  4.34
◦ initial volume of titrated acid: Va [L] = 0.100
◦ chemical formula of titrant base: NaOH
◦ concentration of titrant base: Ctit [molL⁻¹] = 0.010
◦ volume of titrant solution at (first) equivalence point: Veq [L] = 0.010

Find hereafter the corresponding titration curve.
```



https://github.com/philippeloe/moser

ToolBoxLab

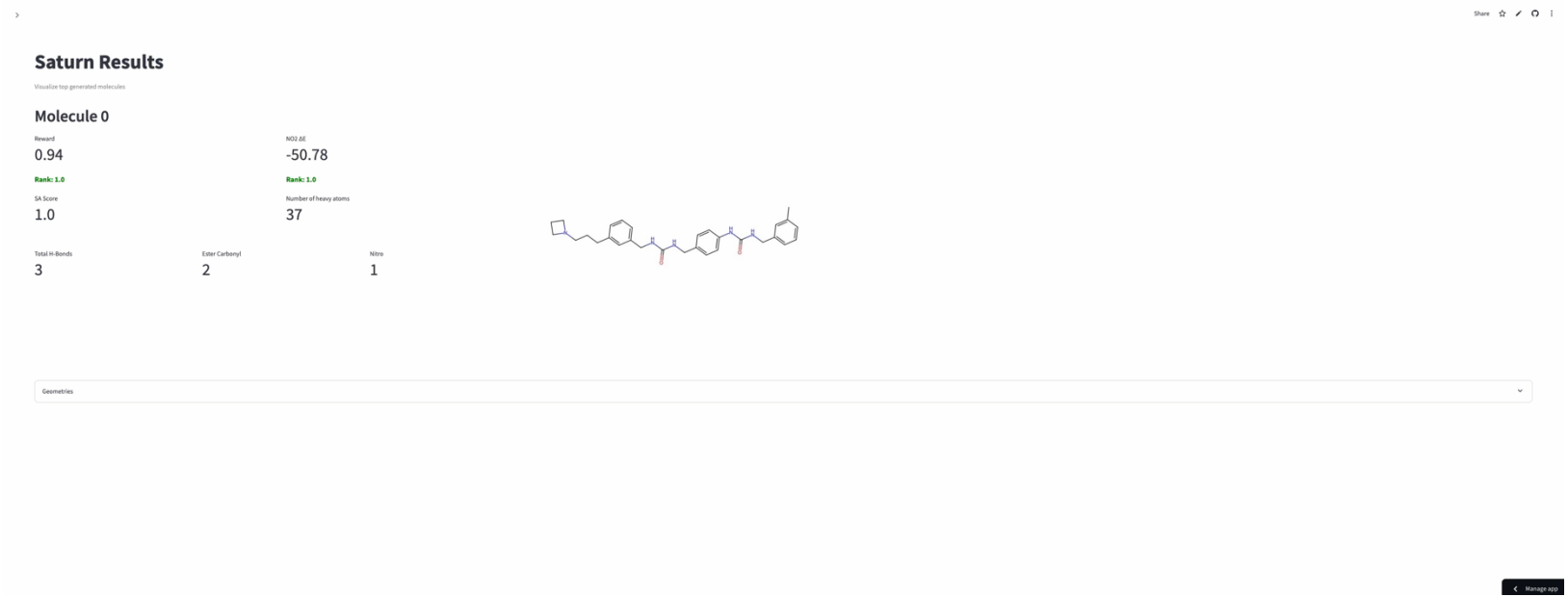




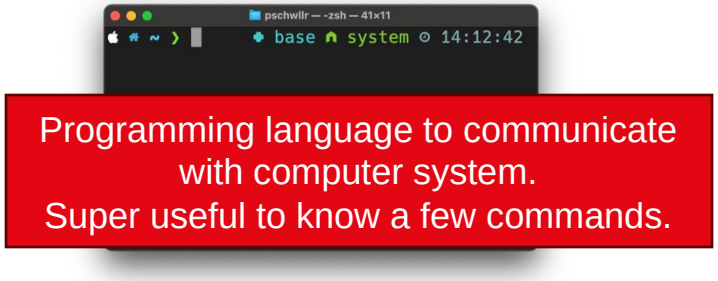
A interactive streamlit app with different tools

https://github.com/Arnautovich/ToolboxLab

Streamlit app from an actual collaboration



The goal for this week – setup your computer



Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage in bash

Visual Studio Code

A better text editor to handle Python code.

Install VS Code as interactive development environment

General purpose programming language.
Create scripts, programs, webapps, …

ANACONDA®

Install Python with Conda
(main course programming language)

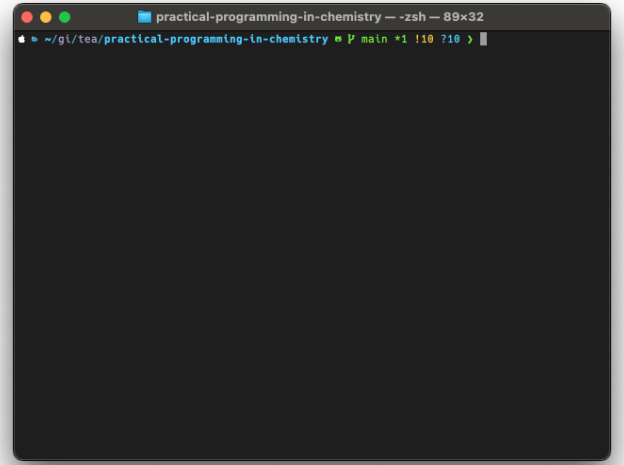
A platform to share, track, and store code.

GitHub

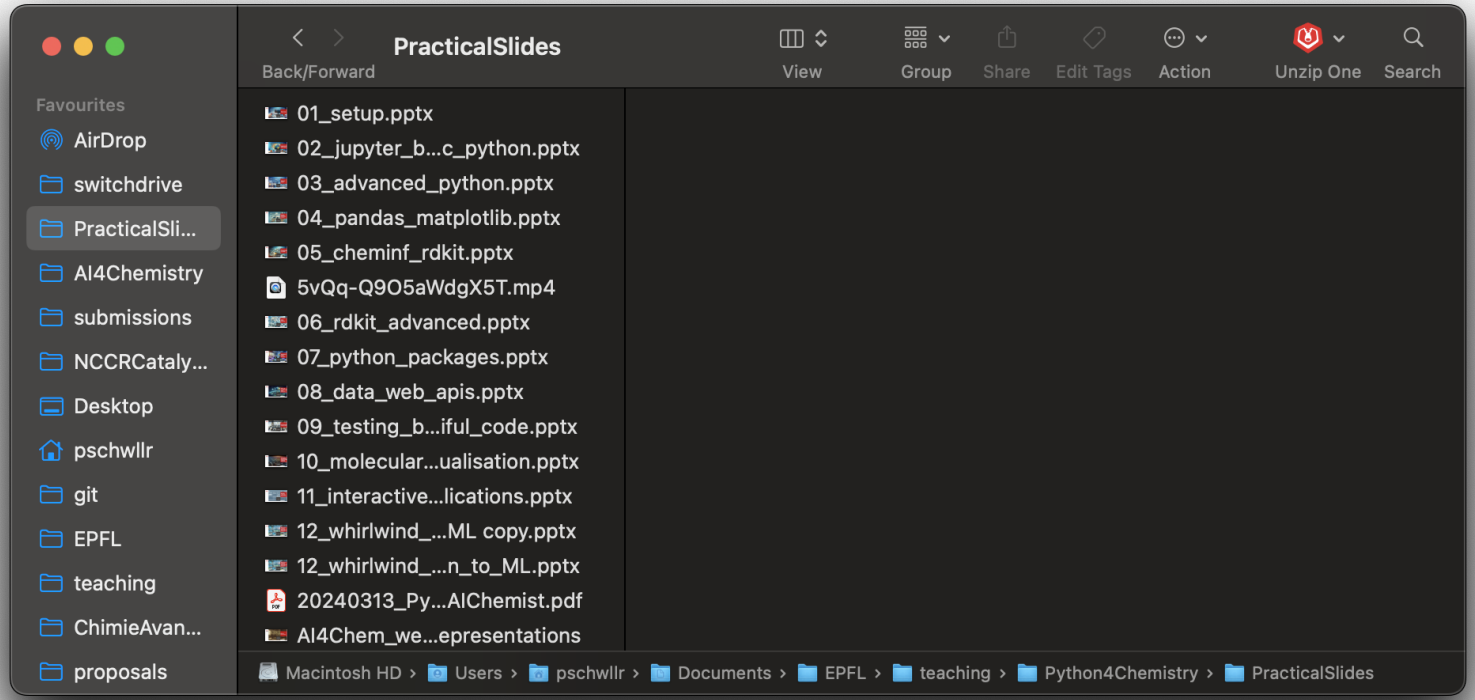
Creating a GitHub account
and getting familiar with basic commands

Using Command line interface (CLI) with bash

- The command line interface (CLI) allows users to **interact with the computer**'s operating system or software by **typing text-based commands**.
- MacOS/Linux → Terminal (bash/zsh)
- Windows → **Git Bash**

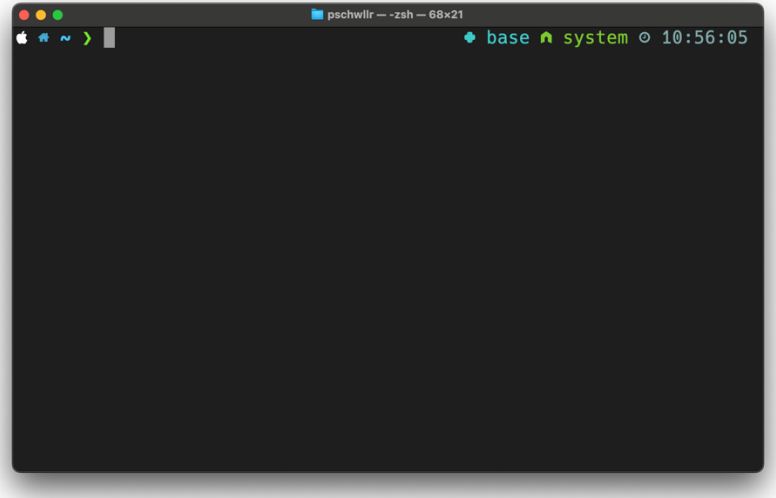


So far, how you navigated through your computer is by clicking through folders



The command line interface gives you the ability to navigate through your computer using commands.

- You can open the interface:
  - Mac: Spotlight (⌘ + Space) and type "**Terminal**"
  - Windows: Start menu and type "**Git Bash**"



Your name

The name of your computer

```
username@computer-name:~$
```

After the $,
is where you can type
commands.

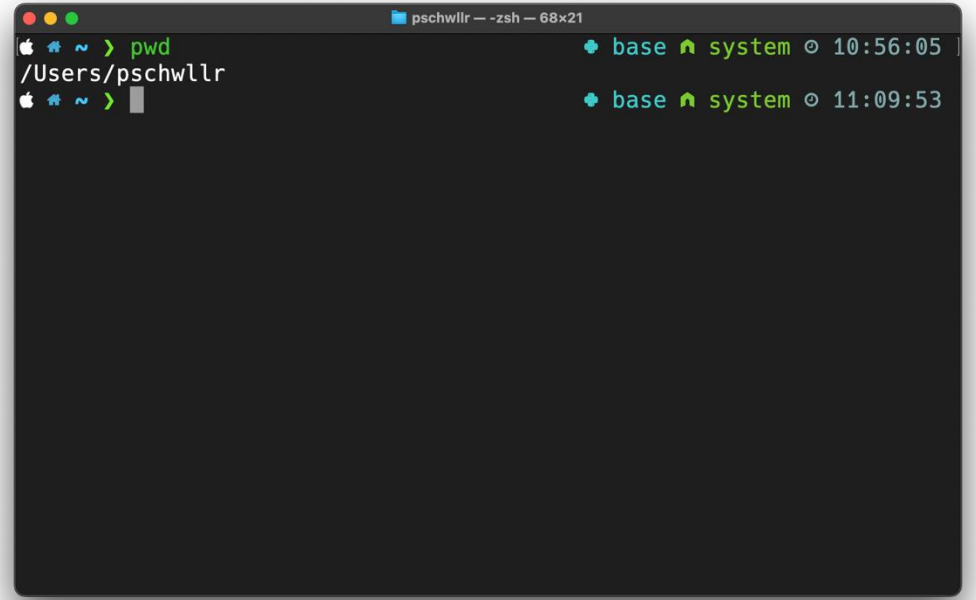
The current folder
- ~ means the home directory
- the starting point when you open a new terminal window

Useful bash commands

bash is the name of the language
for those commands

“pwd” (print current working directory) - where am I currently?

pwd + ENTER → **/users/pschwllr**, which is my **home directory**

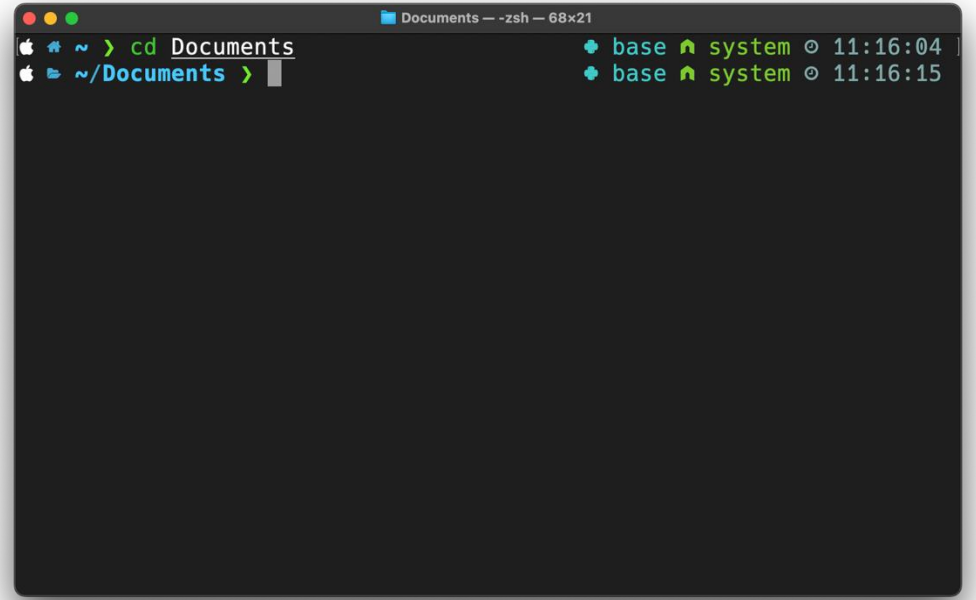


“cd” (change directory) - moving around

```
# Moving around
cd Documents          # Change Directory to Documents
cd ..                 # Go up one level
cd ~                  # Go to home directory
cd Desktop            # Go to Desktop
cd "My Folder"        # Use quotes for names with spaces
```

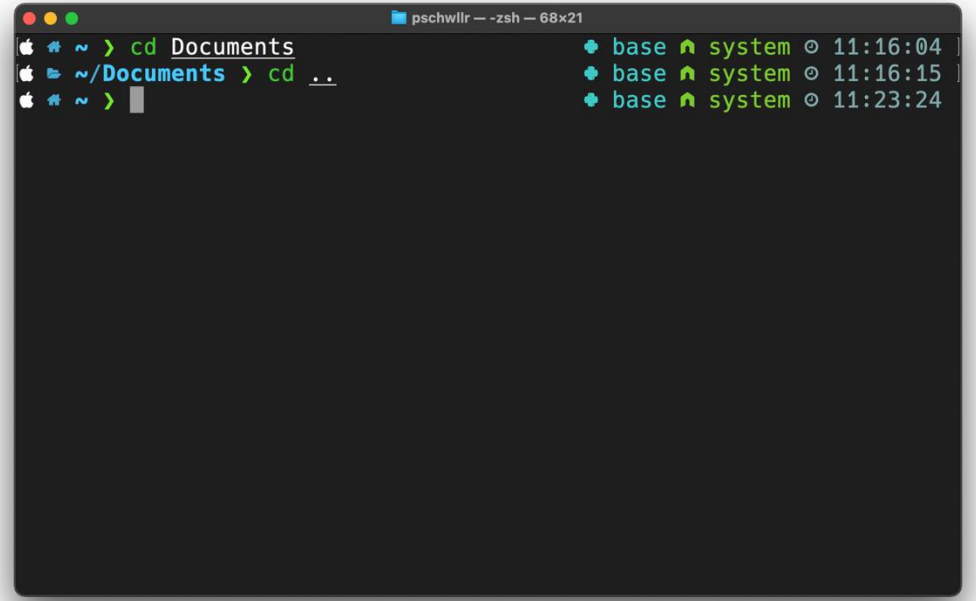
“cd” (change directory) - moving around

```
# Moving around
cd Documents          # Change Directory to Documents
cd ..                 # Go up one level
cd ~                  # Go to home directory
cd Desktop            # Go to Desktop
cd "My Folder"        # Use quotes for names with spaces
```



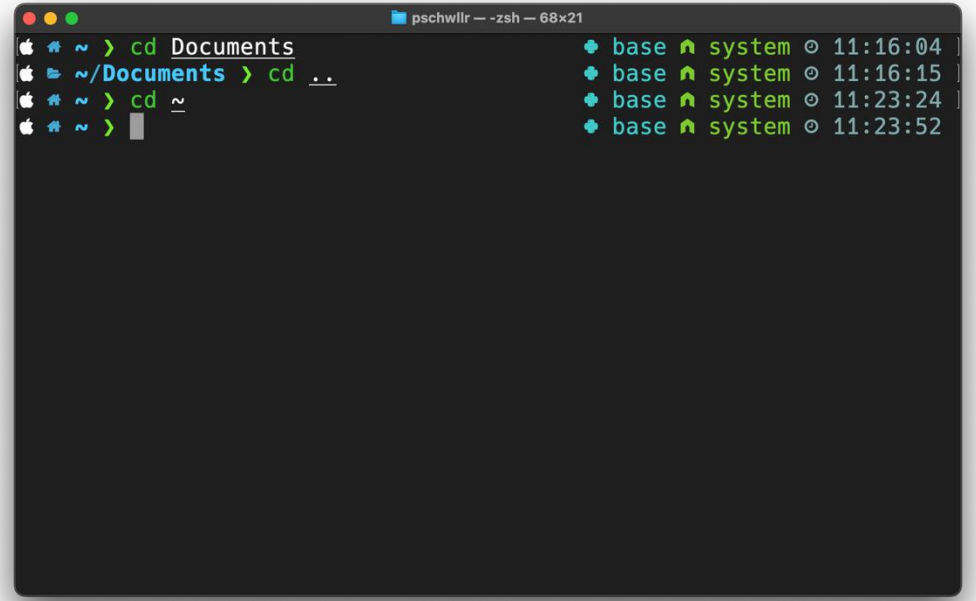
“cd” (change directory) - moving around

```
# Moving around
cd Documents          # Change Directory to Documents
cd ..                 # Go up one level
cd ~                  # Go to home directory
cd Desktop            # Go to Desktop
cd "My Folder"        # Use quotes for names with spaces
```



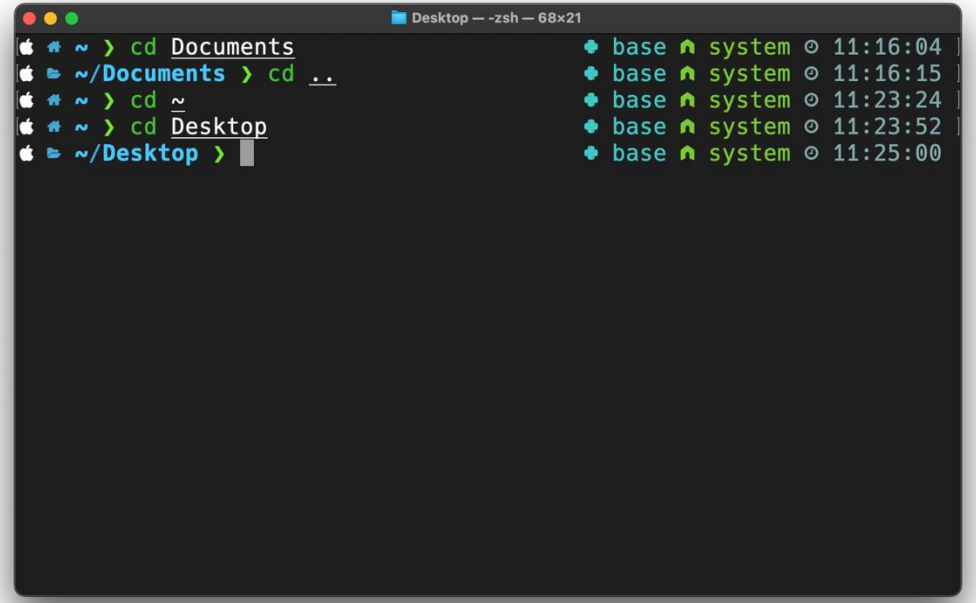
“cd” (change directory) - moving around

```
# Moving around
cd Documents          # Change Directory to Documents
cd ..                 # Go up one level
cd ~                  # Go to home directory
cd Desktop            # Go to Desktop
cd "My Folder"        # Use quotes for names with spaces
```



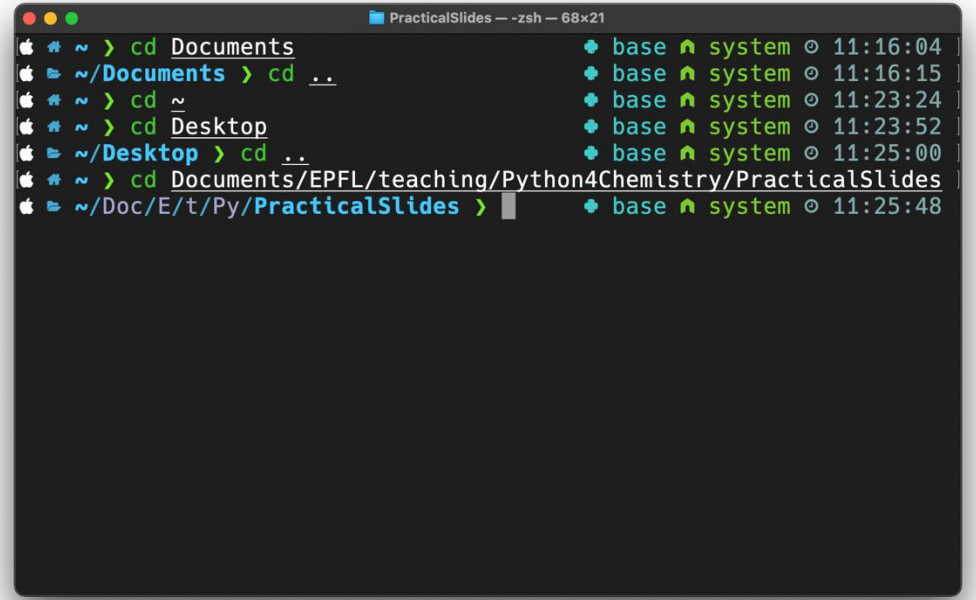
“cd” (change directory) - moving around

```
# Moving around
cd Documents          # Change Directory to Documents
cd ..                 # Go up one level
cd ~                  # Go to home directory
cd Desktop            # Go to Desktop
cd "My Folder"        # Use quotes for names with spaces
```

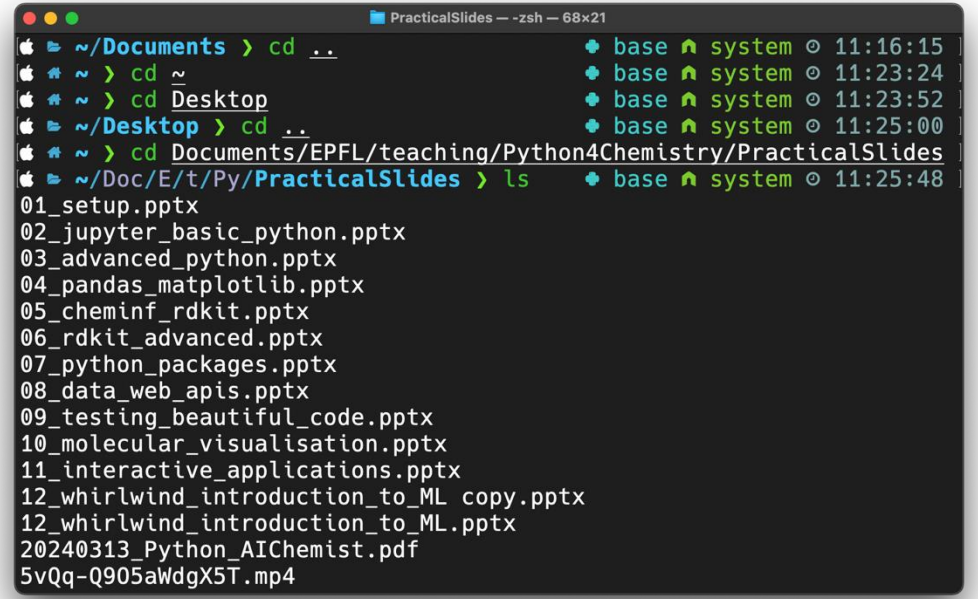


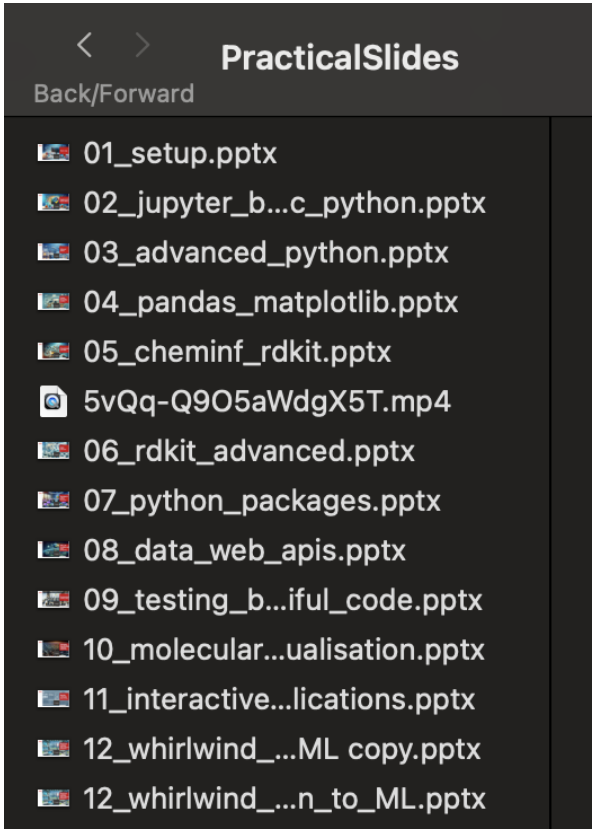
“cd” (change directory) - moving around

```
# Moving around
cd Documents          # Change Directory to Documents
cd ..                 # Go up one level
cd ~                  # Go to home directory
cd Desktop            # Go to Desktop
cd "My Folder"        # Use quotes for names with spaces
```



“ls” (list directory) - what’s in the current folder?



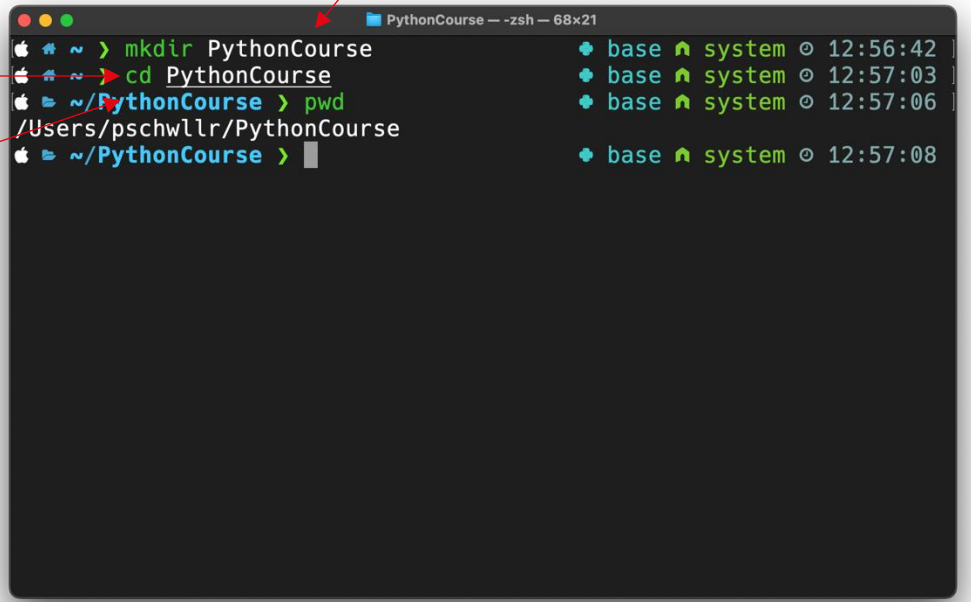


“mkdir” (Make directory)

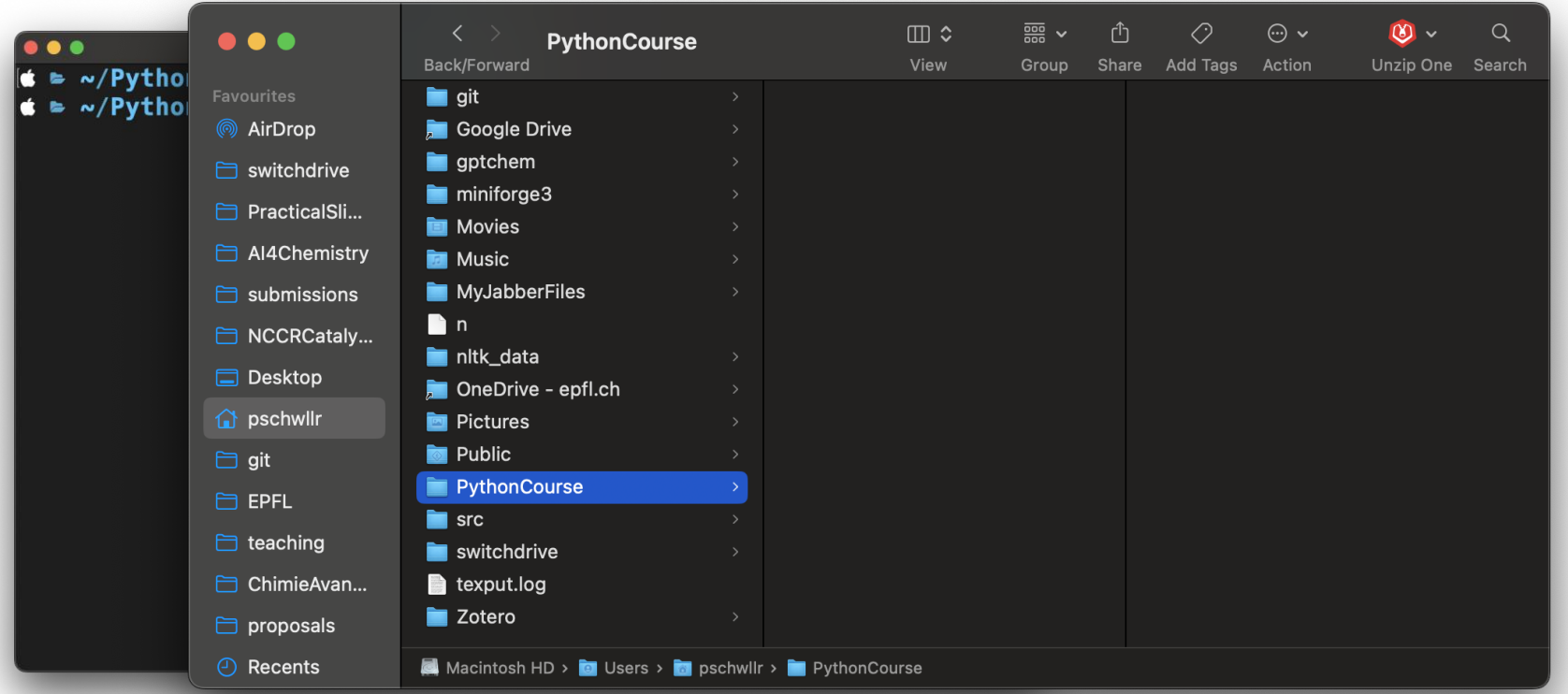
Create a PythonCourse folder

Navigate to the new folder

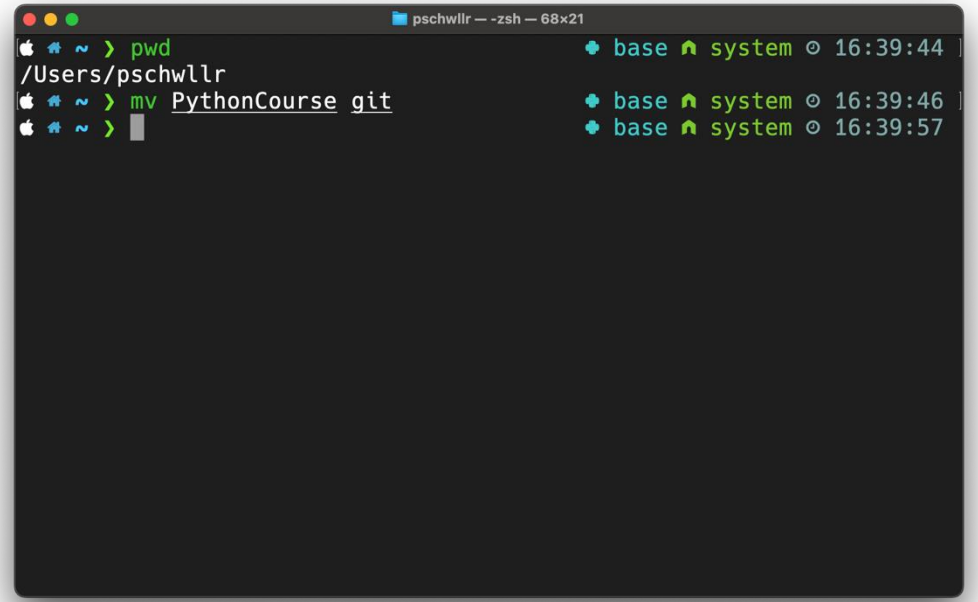
Get the path to the current folder.



“open …”
- open either a folder or file



“mv” (move)
- move file or folder / rename



What are the two things that could happen here?

If the “git” folder exists:
→ It will move “PythonCourse” into “git”

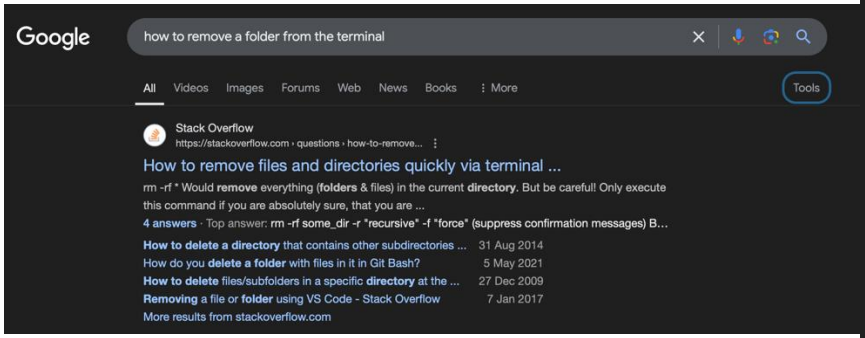
If the “git” folder does not exist:
→ It will rename “PythonCourse” to “git”

“rm” (remove)
- Remove files and directories (be careful)

```
# Creating and removing directories
mkdir ChemLab        # Make a directory named ChemLab
rmdir ChemLab        # Remove empty directory
rm -r ChemLab       # Remove directory and all its contents (be careful!)
```

It’s hard to remember all those commands.

Google/ChatGPT is your friend



how to remove a folder from the terminal

To remove a folder from the terminal, use one of the following commands depending on your needs:

1. Remove an empty folder

bash Copy Edit

```
rmdir folder_name
```

This only works if the folder is empty.

2. Remove a folder and its contents recursively

bash Copy Edit

```
rm -r folder_name
```

This removes the folder and all files/subdirectories inside it.

3. Force delete a folder (dangerous)

bash Copy Edit

```
rm -rf folder_name
```

Important points

- Case sensitivity matters (**Desktop** is different from **desktop**)
- **Spaces in names need quotes** (”My folder”)
- The power (and responsibility) of delete commands
- **Tab completion** to avoid typing errors
- **Using up/down arrow keys** to recall previous commands

Bash is even more powerful

Additional commands to work with files

```
# Creating and viewing files
touch data.txt          # Create empty file
cat data.txt           # Display file contents
less data.txt          # View long files (page by page)
head -n 5 data.txt     # Show first 5 lines
tail -n 5 data.txt     # Show last 5 lines

# Copying and moving
cp data.txt backup.txt          # Copy a file
cp -r DataFolder BackupFolder   # Copy a directory and contents
mv old.txt new.txt             # Rename/move a file
mv data.txt ../DataFolder/     # Move file to different directory

# Finding things
find . -name "*.txt"           # Find all .txt files in current directory
grep "pH" data.txt             # Search for "pH" in file
grep -r "glucose" .            # Search in all files recursively

# File information
file data.txt          # Show file type
wc -l data.txt        # Count lines in file
du -h data.txt        # Show file size
```

Terminal demo?

System and help


```
# Getting help
man ls                  # Show manual for ls command
ls --help               # Quick help for ls
history                 # Show command history

# System information
date                    # Show current date/time
whoami                  # Show current user
df -h                   # Show disk space usage
top                     # Show running processes
```

Bash / Python comparison

- **Bash is a command-line shell language** that's primarily designed to:
  1. Navigate and manipulate files/folders
  2. Run programs and chain them together
  3. Automate system tasks
- **Python is a general-purpose programming language** that's great for:
  1. Data analysis and visualization
  2. Scientific computing
  3. Writing complex algorithms
  4. Creating applications

```python
# Python script to do the actual analysis
import pandas as pd
import numpy as np

def analyze_spectrum(data):
    # Complex mathematical operations
    # Statistical analysis
    # Machine learning
    # Plotting results
```

```bash
# Bash script to prepare data for analysis
for file in *.raw; do
    # Convert raw instrument files to CSV
    converter "$file" "${file%.raw}.csv"
done

# Now run the Python analysis
python analyze_spectra.py
```

Use Python

To run this Python script.

The goal for this week – setup your computer

Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage in bash

Visual Studio Code

A better text editor to handle Python code.

Install VS Code as interactive development environment

General purpose programming language.
Create scripts, programs, webapps, …

ANACONDA®

Install Python with Conda
(main course programming language)

A platform to share, track, and store code.

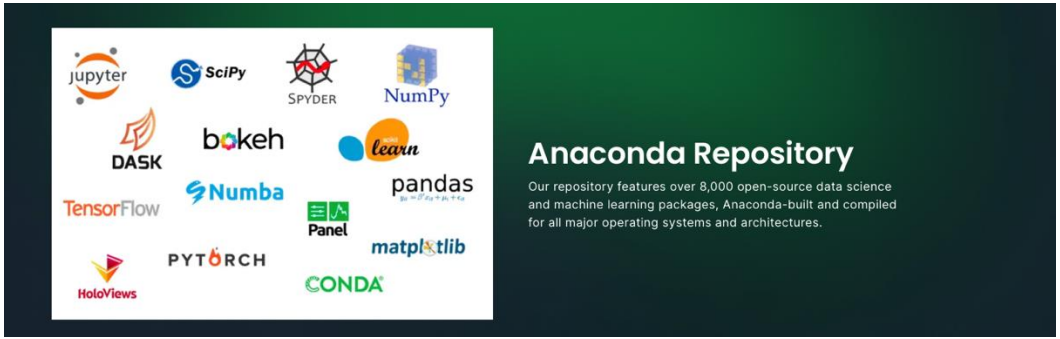
GitHub

Creating a GitHub account and getting familiar with basic commands

Anaconda – Python distribution

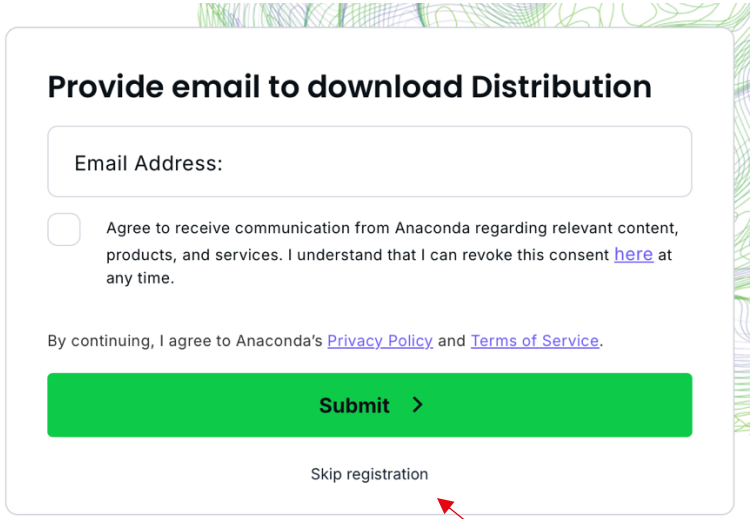
- Designed for scientific computing, data science, and machine learning.
- Gives you access to thousands of pre-packaged modules
- Simplified **package management system**
- **Environment management**

You can download it from: https://www.anaconda.com/download



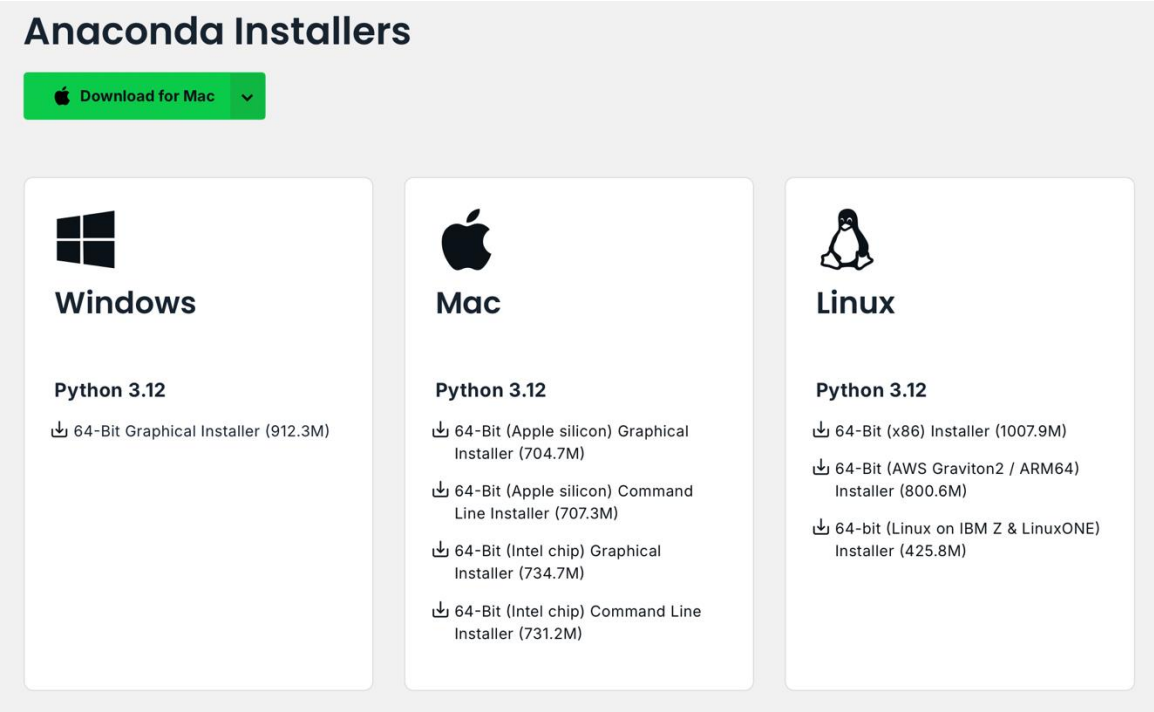
ANACONDA®

Anaconda – Python distribution



Skip registration

Anaconda – Python distribution



Note: You will have time to do that in the exercise session.
If you are unsure, which one, we are happy to help.
Wait until the exercises!

Python ecosystem

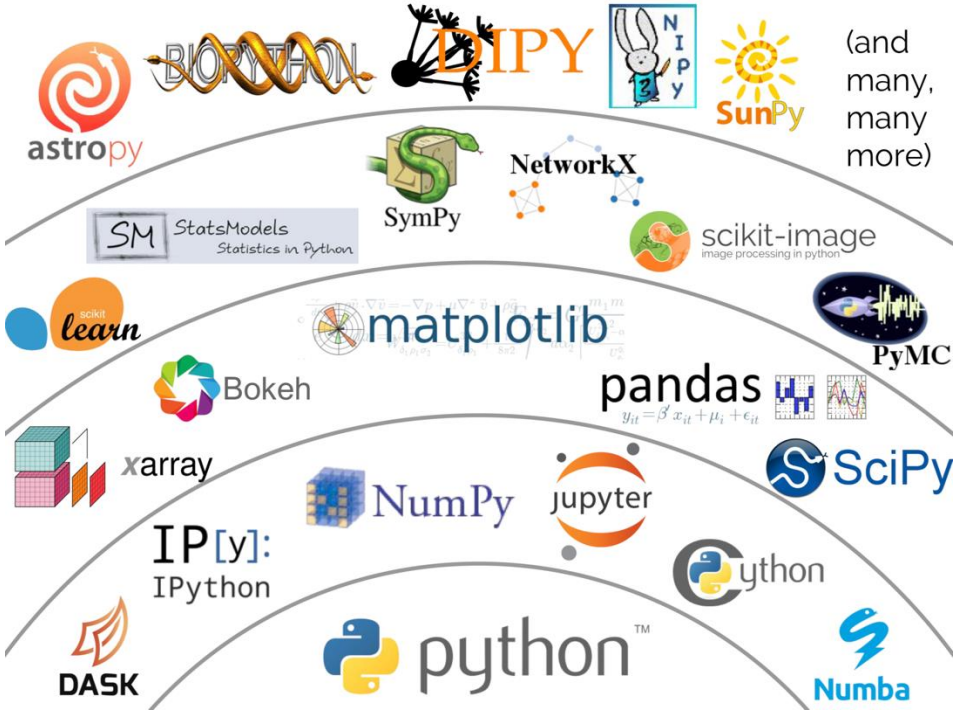


Image: https://blueskyproject.io

The goal for this week – setup your computer

Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage in bash

Visual Studio Code

A better text editor to handle Python code.

Install VS Code as interactive development environment

General purpose programming language.
Create scripts, programs, webapps, …

ANACONDA®

Install Python with Conda
(main course programming language)

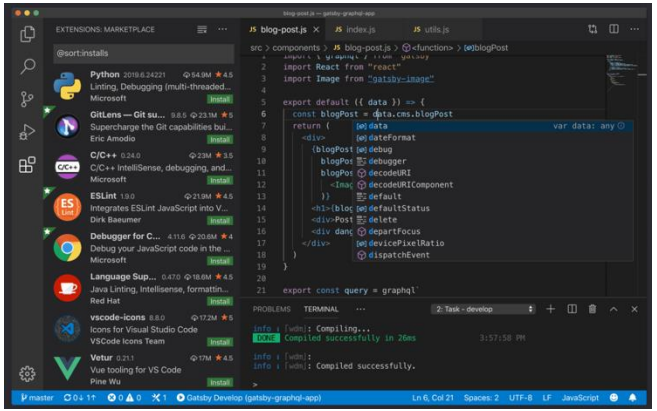
A platform to share, track, and store code.

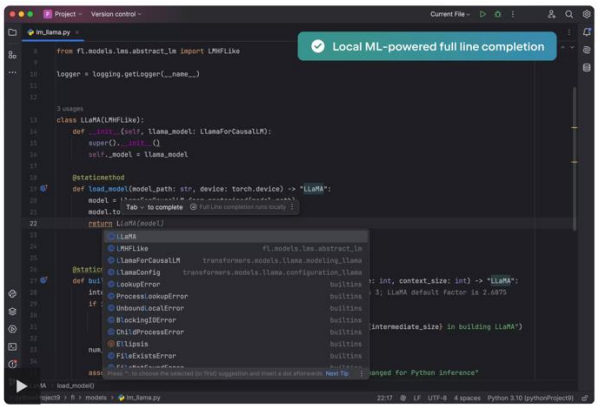
GitHub

Creating a GitHub account
and getting familiar with basic commands

Where to write code? In an interactive development environment (IDE)

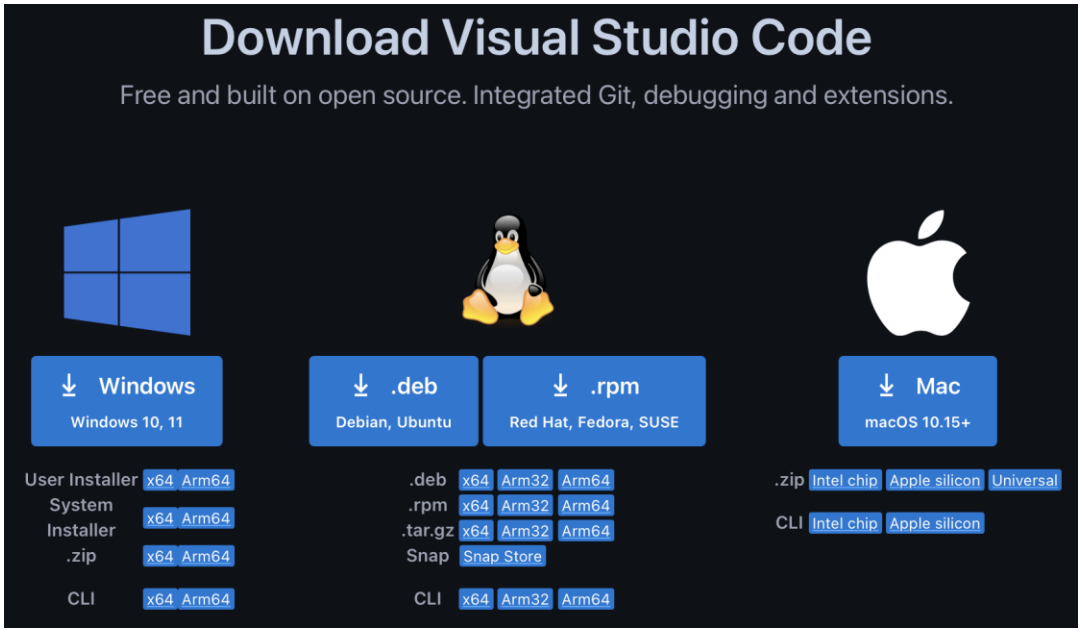
- You could use a **text editor**.
- But IDEs provide useful tools for **coding, error checking, autocompletion, etc…**
- Popular examples are **VS Code**, PyCharm, and Jupyter lab





Here we will use VS Code (you most likely already have it from last year)

- https://code.visualstudio.com/download



The goal for this week – setup your computer

Programming language to communicate with computer system.
Super useful to know a few commands.

Basic command line usage in bash

Visual Studio Code

A better text editor to handle Python code.

Install VS Code as interactive development environment

General purpose programming language.
Create scripts, programs, webapps, …

ANACONDA®

Install Python with Conda
(main course programming language)

A platform to share, track, and store code.

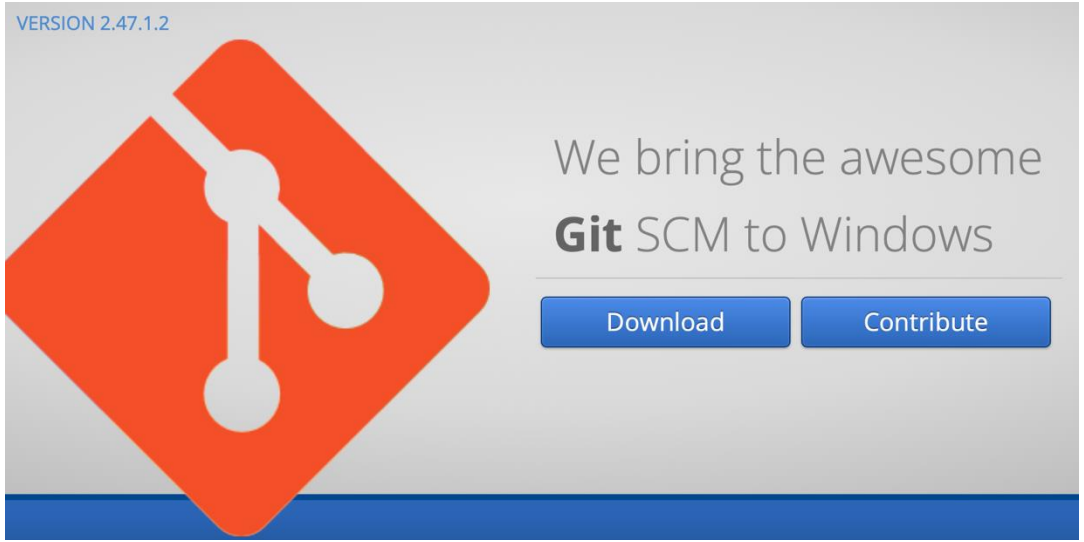
GitHub

Creating a GitHub account
and getting familiar with basic commands

Windows users only → Git for Windows

- https://gitforwindows.org
- Download and install it (if you have not already done so)
- Added benefit
  → Git bash

- Mac users should have git natively installed.



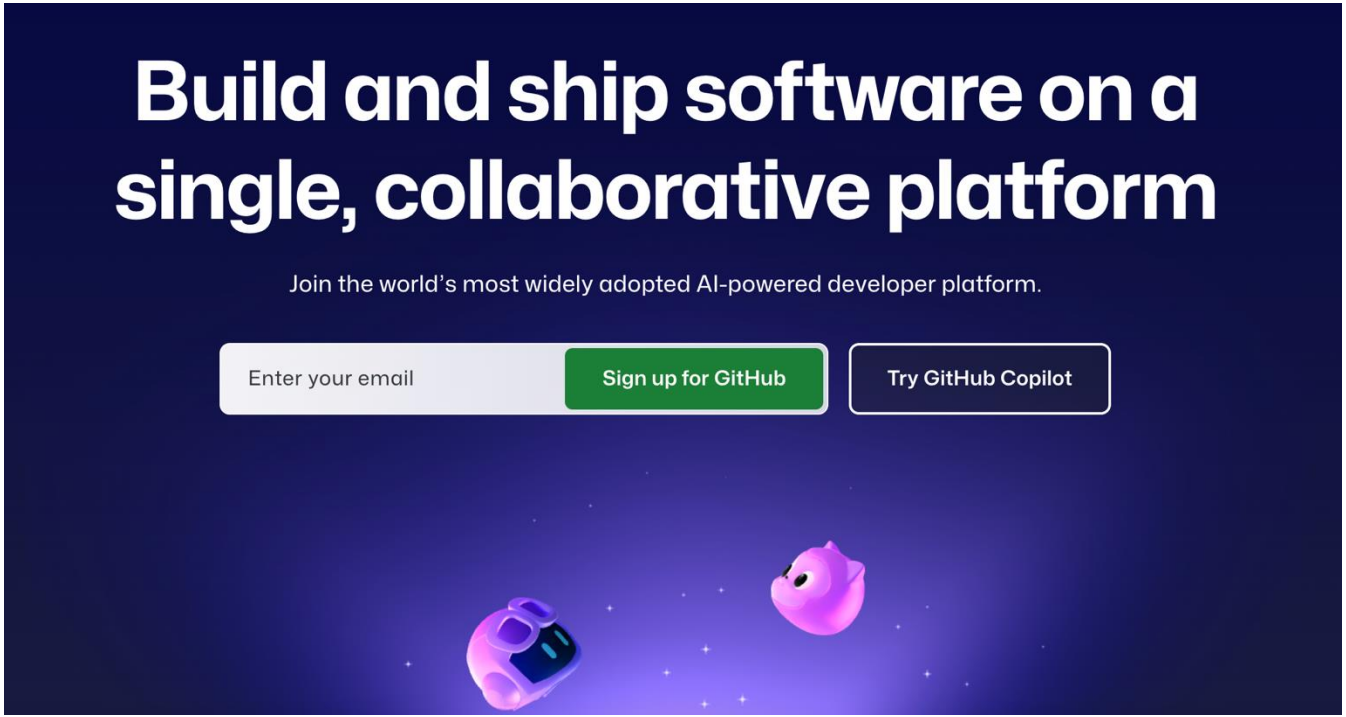
GitHub – the platform we will use to share the exercises

Git is a Version Control System (VCS) ideal for
• tracking changes,
• collaborating,
• sharing in coding projects,
allowing users to revert to older versions and work together seamlessly.

GitHub (based on Git) offers an online platform with an easy-to-use interface. Almost a **social platform for coding**.

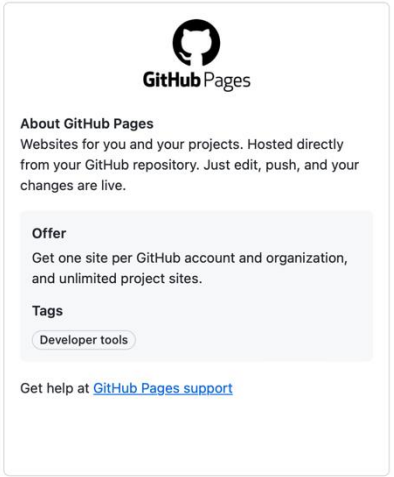
https://github.com/schwallergroup/practical-programming-in-chemistry-exercises

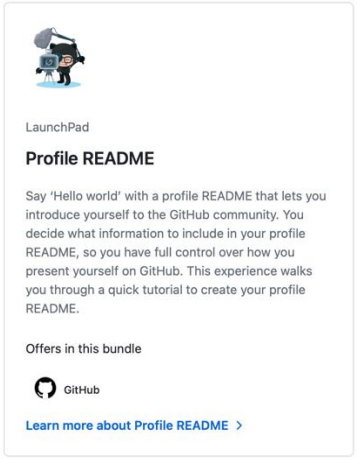
Sign up for it. → github.com



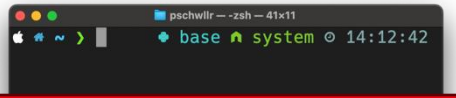
Student Developer pack

- https://education.github.com/pack
- Give you free access to unlimited private repositories (and other things people pay for usually)





The goal for this week – setup your computer



Getting to know
the command line and bash.

Basic command line usage
in bash

Visual Studio Code

Installing VS Code

Install VS Code as interactive
development environment

Downloading Anaconda

ANACONDA®

Install Python with Conda
(main course programming language)

Making a GitHub account

GitHub

Creating a GitHub account
and getting familiar with basic commands

Most of the exercises, are just making sure, you are properly set up.

And then, there is a game to learn bash.

Game to learn bash (bashcrawl)

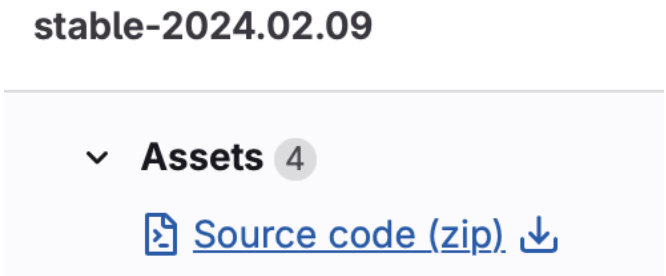
Download and unpack

If you haven't downloaded Bashcrawl yet, go to https://gitlab.com/slackermedia/bashcrawl/-/releases and download the `Source code (zip)` file.

Once the ZIP file has been downloaded to your computer, unzip it as usual.

Now it's time to try it out on your computer.

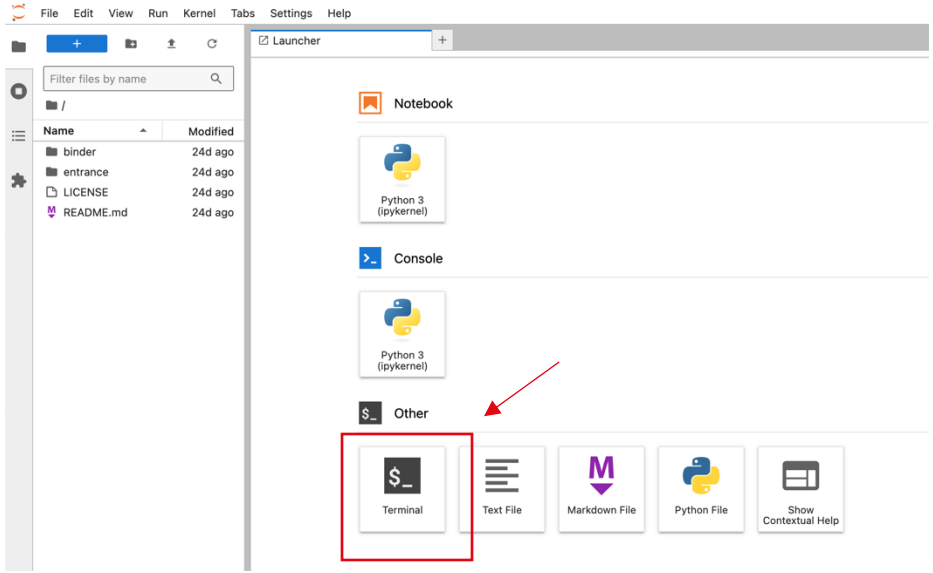
https://gitlab.com/slackermedia/bashcrawl



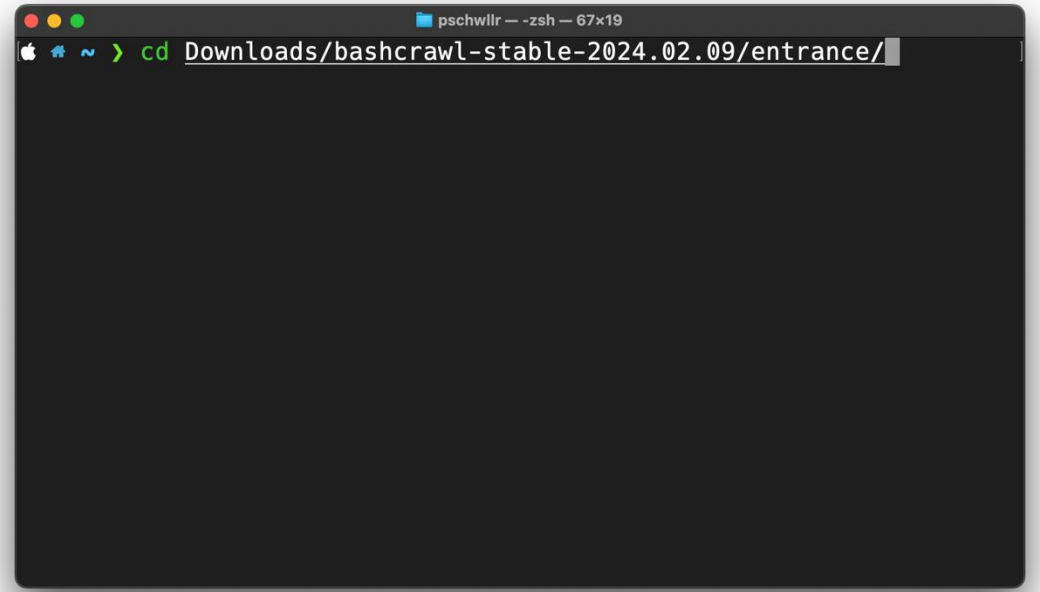
https://gitlab.com/slackermedia/bashcrawl/-/archive/stable-2024.02.09/bashcrawl-stable-2024.02.09.zip

Alternative launch it online with binder (just click on the link and then, the terminal)

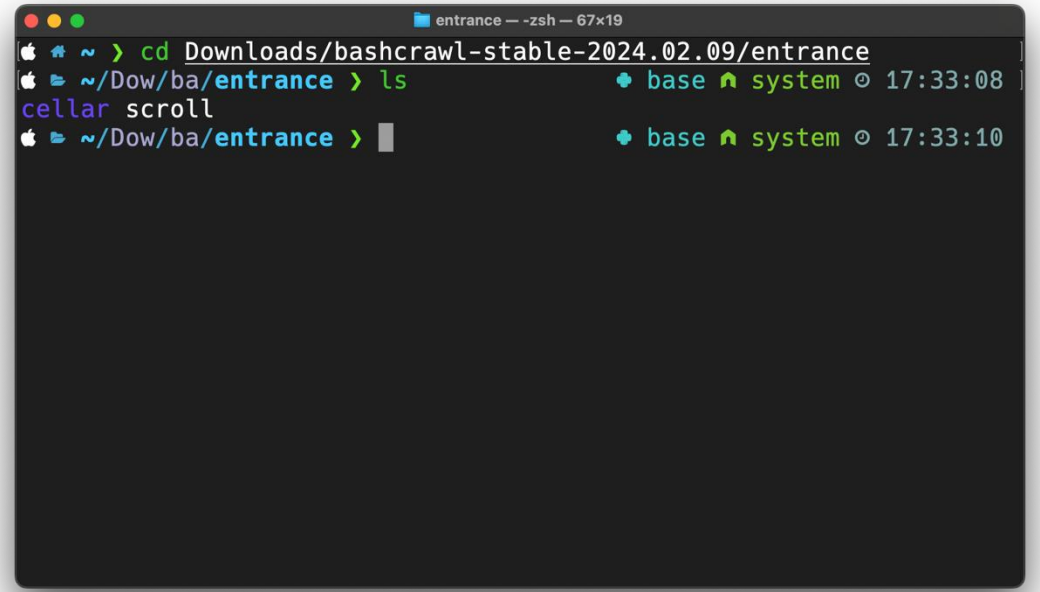
- https://mybinder.org/v2/gl/nthiery%2Fbashcrawl/HEAD



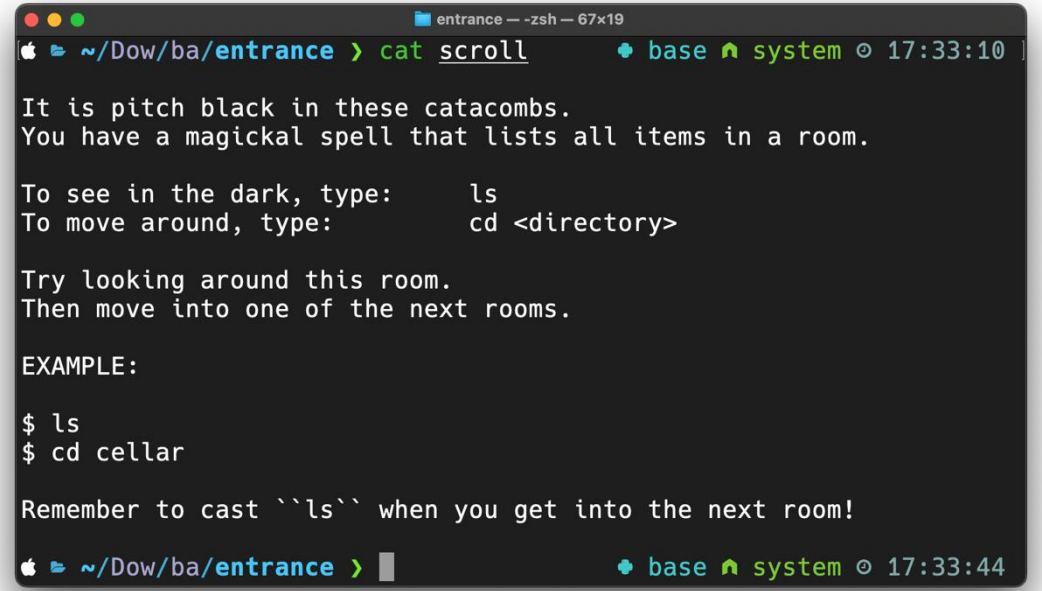
After the download, you can navigate to the entrance with cd



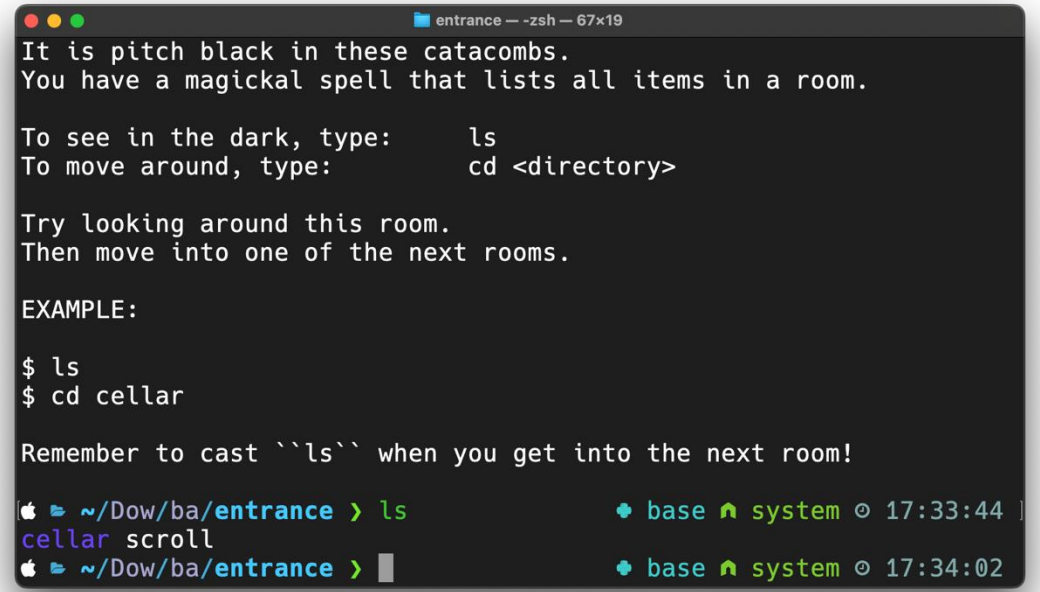
ls will list all files in the folder



Every room has a text file called “scroll”



With “ls” you see that there is a room (or folder) called cellar.



“cd cellar”, then “cat scroll”, and you have the next instructions

```
cellar — -zsh — 67×25
~/Dow/ba/entrance ❯ cd cellar          base system 17:34:02
~/Dow/ba/e/cellar ❯ cat scroll         base system 17:34:44

# Illusions are strong here.
# It is difficult to tell what is a doorway and what is an object.
#
# The magic spell you use to look can be augmented.
#
# From now on, cast your spell like this:
#
# ls -F
#
# Directories (the rooms of these catacombs) end with a / symbol.
#
# Encounters (programs) end with a * symbol.
#
# You can avoid having to type `ls -F` every time by running the
# following:
#
# alias ls='ls -F'
#
# This is known as a shell or command alias.  With this alias,
# typing simply ls by itself will run ls -F. Try it out!
#
~/Dow/ba/e/cellar ❯                    base system 17:34:48
```

```
armoury — -zsh — 67×25
# can come back to this treasure later!


~/Dow/ba/e/cellar ❯ cd armoury         base system 17:38:21
~/Dow/ba/e/c/armoury ❯ ls              base system 17:38:26
chamber/        potion*         scroll          treasure*
~/Dow/ba/e/c/armoury ❯ ./potion        base system 17:38:26
You have found a potion bottle of swirling
green liquid. Do you want to drink it?  y/n

y
The taste of a rustic green plant fills your mouth.  It
warms and strengthens you.

Create a variable for your health points (HP).  You have
15HP:

export HP=15

You can check your health at any time:

echo $HP

~/Dow/ba/e/c/armoury ❯ export HP=15
~/Dow/ba/e/c/armoury ❯                 base system 17:38:47
```

It is fun to explore that cellar, and run the bash commands that are suggested.

Now, there will be a short break of 5 minutes, and then, the TA will introduce the exercises.